Managing the Performance of Large, Distributed Storage Systems

Scott A. Brandt

and

Carlos Maltzahn, Kleoni Iouannidou, Anna Povzner,
Roberto Pineiro, Andrew Shewmaker, and Tim Kaldewey
University of California Santa Cruz

and

Richard Golding and Ted Wong, IBM Almaden Research Center

and

Darren Sawyer, NetApp

HEC FSIO 2011

What types of QoS guarantees are useful to HEC application scientists (esp. at exascale)?

- Isolation / virtualization of storage performance
- Guaranteed
  - Checkpoint performance
  - Data capture
  - Performance under varying workloads
  - Performance under concurrent workloads
  - Performance under failure
  - Performance for viz
- What guarantees: throughput & latency

UC SANTA CRUZ

ISSDM

What is the relationship between QoS infrastructure and other job scheduling infrastructure (e.g., batch schedulers, MapReduce scheduling)?

- Very close relationship
- QoS requirements must inform all system schedulers
- Only possible if
  - Each resource is well understood
  - Requirements are well understood
  - Simplifying principles are found and used

Is the ratio of funded to productized work in QoS lower than in other areas? Yes. If so, why?

- New area—focus has traditionally been on performance, performance, and performance
- Hard *and* requires new ways of thinking
- Crosscutting: storage, real-time, distributed systems, networking, scheduling, ...
- Enabled by excess processor capacity
- Needed by cloud computing and virtualization
- Starting to make it into commercial products
- More on the horizon...

Challenges

- Legacy (intransigent) applications and users
- Scaling—aggregate management
- Crossing the threshold to usability
- Varied resources, applications, workloads
- Interference between I/O streams
- System management tasks

Distributed systems need performance guarantees

- *Many* systems and applications want I/O performance guarantees
  - Multimedia, high-performance simulation, transaction processing, virtual machines, cloud services, service level agreements, real-time data capture, sensor networks, ... system tasks like backup and recovery ... even so-called best-effort applications
- Providing guarantees is difficult
  - Interacting resources, dynamic workloads, interference among workloads, non-commensurable metrics
- Needs
  1. Guaranteed performance
  2. Isolation between workloads
  3. High performance

In a nutshell

- Big distributed systems
  - Serve many users/jobs
  - Process petabytes of data
- Data center design
  - Use rules of thumb
  - Over-provision
  - Isolate

- Ad hoc performance management approaches creates *expensive* and *marginal* solutions
- A better system guarantees each user the performance they need from the CPUs, memory, disks, and network

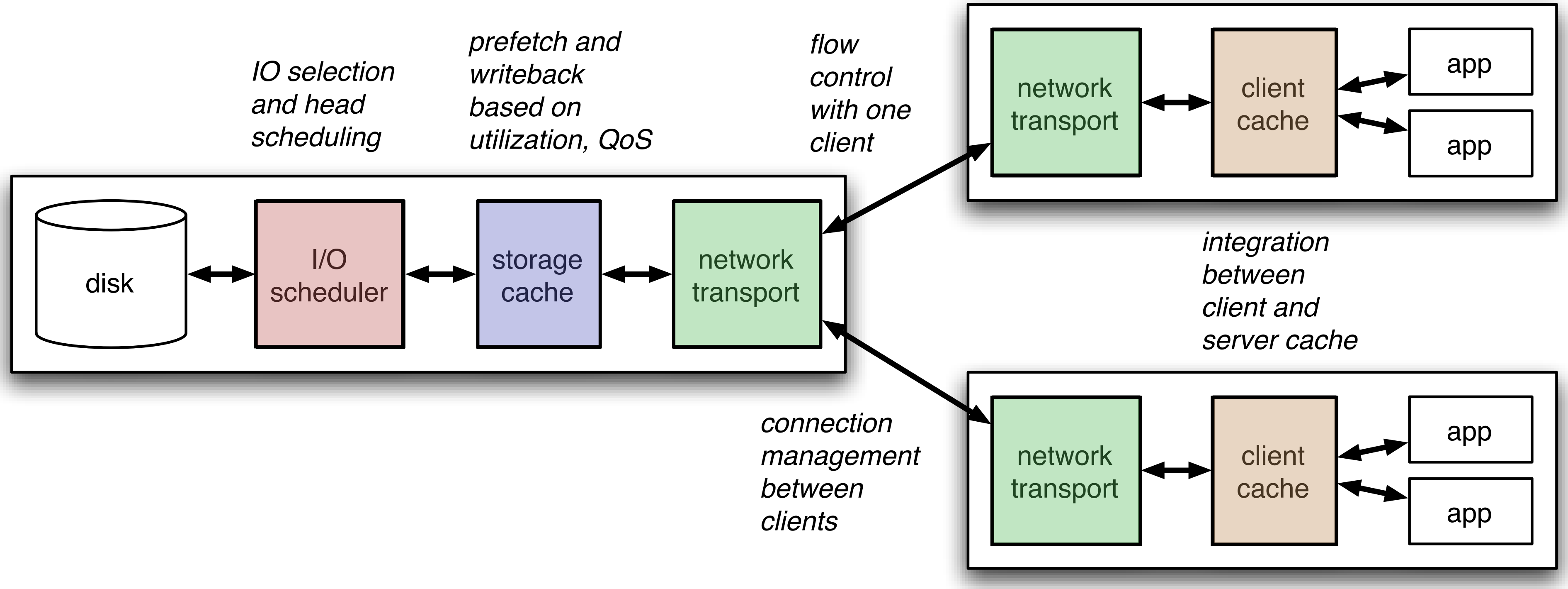
UC SANTA CRUZ

ISSDM

Our approach

1. A uniform model for performance management
2. Apply it to each resource
3. Integrate the solutions

1. Disk I/O
2. Server cache
3. Flow control across network
4. Client cache



Achieving robust guaranteeable resources

- Goal: Unified resource management algorithms capable of providing
  - Good performance
  - Arbitrarily hard or soft performance guarantees with
    - Arbitrary resource allocations
    - Arbitrary timing / granularity
  - Complete isolation between workloads
  - All resources: CPU, disk, network, server cache, client cache

➡Virtual resources indistinguishable from “real” resources with fractional performance

Isolation is key

- CPU
  - 20% of a 3 Ghz CPU should be indistinguishable from a 600 Mhz CPU
  - Running: compiler, editor, audio, video
- Disk
  - 20% of a disk with 100 MB/second bandwidth should be indistinguishable from a disk with 20 MB/second bandwidth
  - Serving: 1 stream, *n* streams, sequential, random

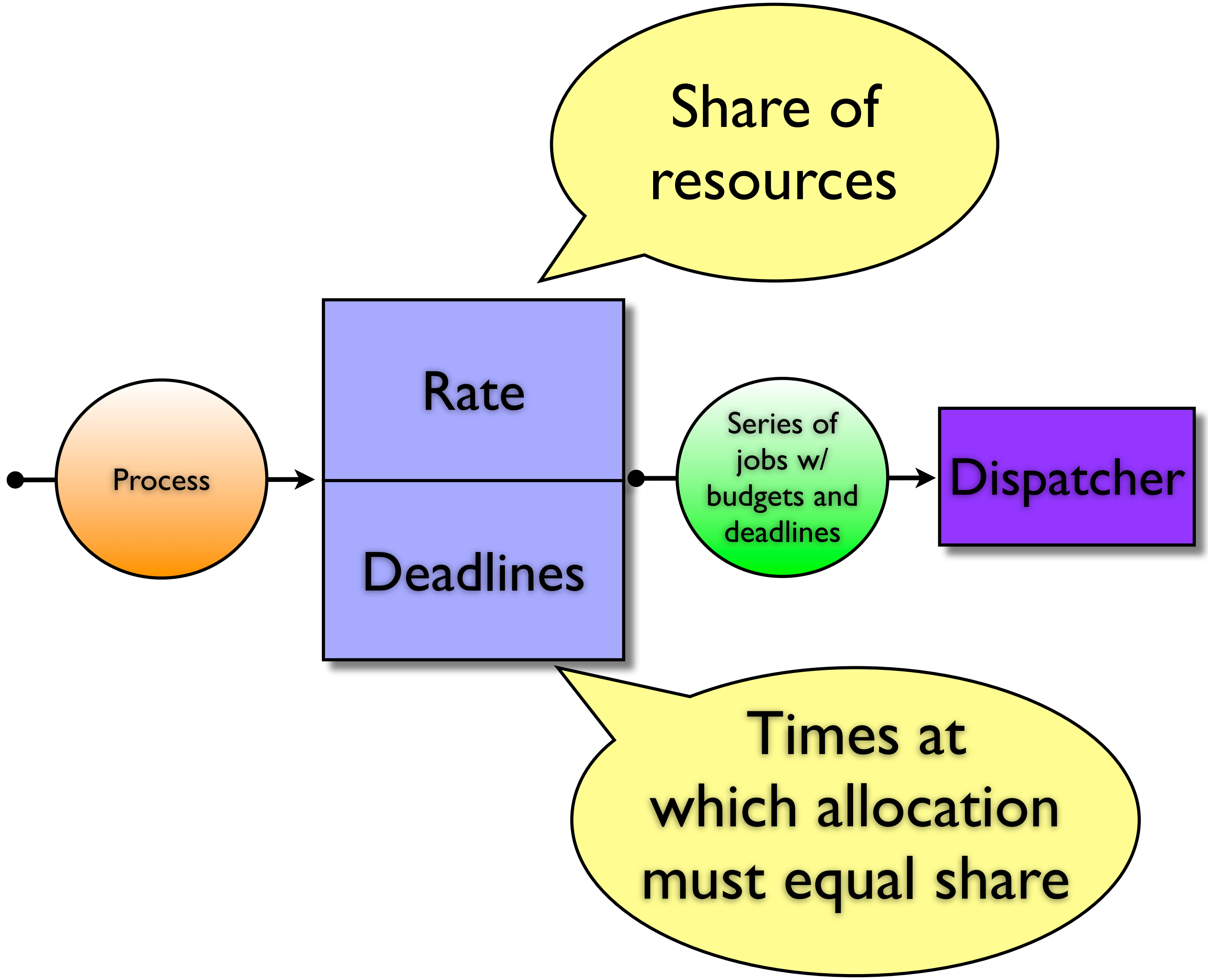
Scott's epistemology of virtualization

- Virtual Machines and LUNs provide good HW virtualization
- Question: Given perfect HW virtualization, *how can a process tell the difference between a virtual resource and a real resource?*
- Answer: By not getting its **share** of the resource **when** it needs it

Observation

- Resource management consists of two distinct decisions
  - Resource Allocation: *How much* resources to allocate?
  - Dispatching: *When* to provide the allocated resources?
- Most resource managers conflate them
  - Best-effort, proportional-share, real-time

The resource allocation/dispatching (RAD) scheduling model



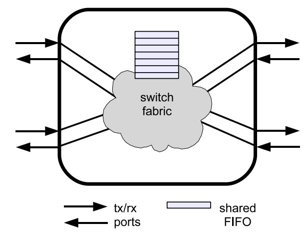
UC SANTA CRUZ

ISSDM

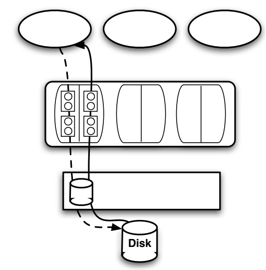
Adapting RAD to disk, network, and buffer cache

- Fahrrad—Guaranteed disk request scheduling
  Anna Povzner

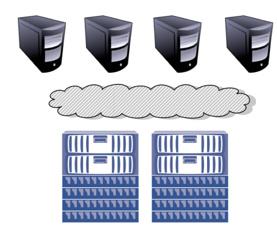
- RADoN—Guaranteeing storage network performance
  Andrew Shewmaker



- Radium—Buffer management for I/O guarantees
  Roberto Pineiro



- Horizon—I/O management for distributed storage systems
  Anna Povzner



UC SANTA CRUZ

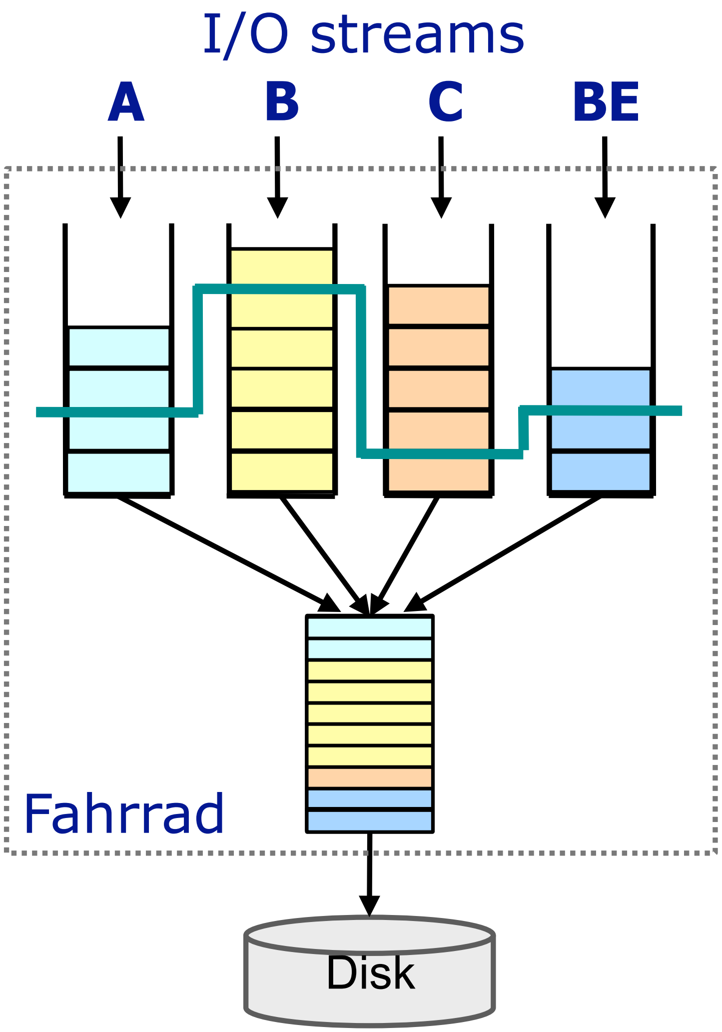
ISSDM

Guaranteed disk request scheduling

- Goals
  - Hard and soft performance guarantees
  - Isolation between I/O streams
  - Good I/O performance
- Challenging because disk I/O is:
  - Stateful
  - Non-deterministic
  - Non-preemptable, and
  - Best- and worst-case times vary by 3–4 orders of magnitude

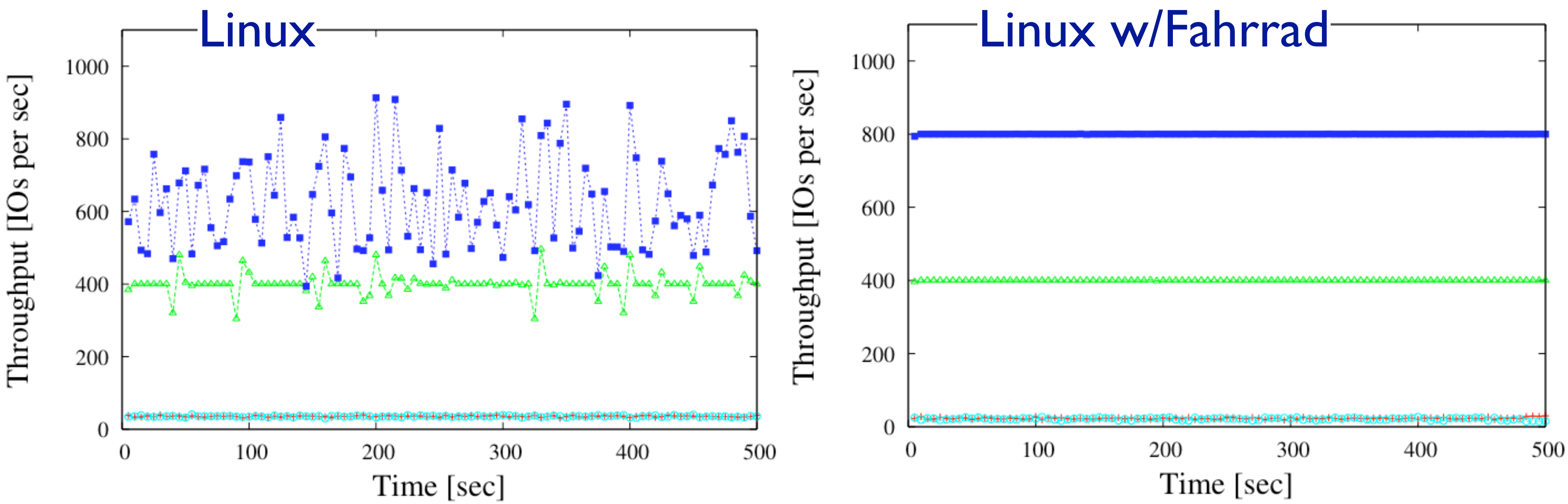
Fahrrad

- Manages disk *time* instead of disk *throughput*
- Adapts RAD/RBED to disk I/O
- Reorders aggressively to provide good performance, without violating guarantees



Anna Povzner, Tim Kaldewey, Scott A. Brandt, Richard Golding, Theodore Wong, and Carlos Maltzahn, "*Efficient Guaranteed Disk Request Scheduling with Fahrrad*", Eurosys 2008.

Fahrrad outperforms Ext2/3



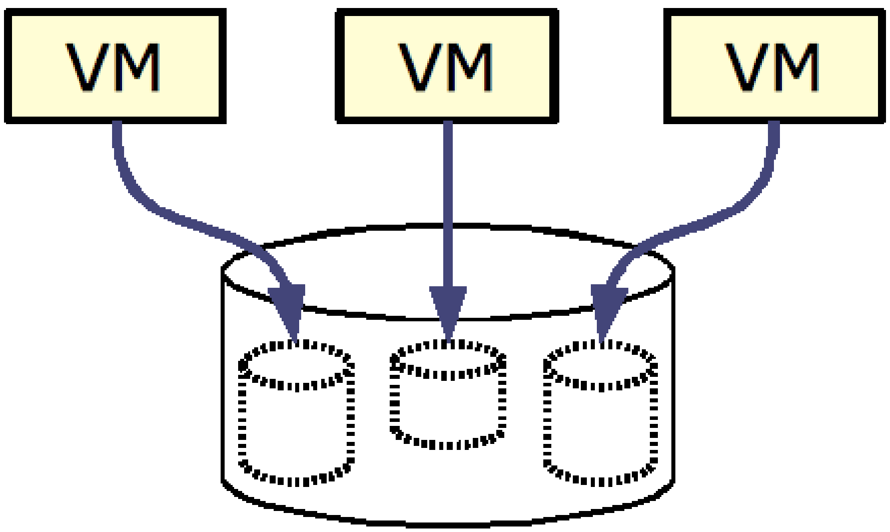
- Workload
  - Media 1: 400 sequential I/Os per second (20%)
  - Media 2: 800 sequential I/Os per second, (40%)
  - Transaction: short bursts of random I/Os at random times (30%)
  - Background: random (10%)
- Result: Better isolation AND better throughput

UC SANTA CRUZ

ISSDM

Fahrrad virtual disks

- Provide workload-independent performance guarantees
- Isolate from other workloads concurrently accessing the device



- LUNs virtualize *storage capacity*
- Fahrrad virtualizes *storage performance*

Fahrrad virtual disks

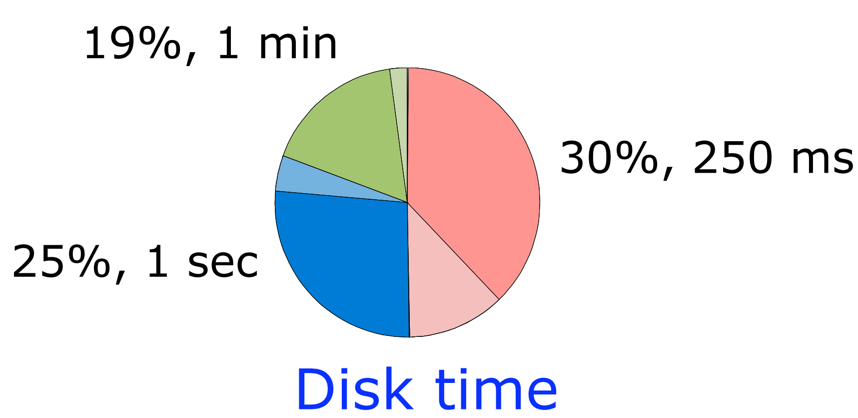
- Implemented with the Fahrrad real-time I/O scheduler
- Guarantee reserved and isolated share of the time on storage device
  - Hard guarantees on performance isolation
  - Virtual disk throughput same as equivalent standalone throughput
- Amount of data transferred:
  - $\forall i,\ D_i(\underbrace{x\%}_{\text{Share of disk}}, \underbrace{t}_{\text{Time}}) = D_i(\underbrace{100\%}_{\text{Share of disk}}, \underbrace{x\% \cdot t}_{\text{Time}})$

UC SANTA CRUZ

ISSDM

Ensuring isolation

- Virtual disk reservation: *disk share (utilization)* and *time granularity (period)*
  - Account for all extra (inter-stream) seeks
  - Reserve overhead utilization to do them
  - Charge each I/O stream for all of the time it uses, including inter- and intra-stream seeks
  - Reservation = Disk Share + Overhead utilization

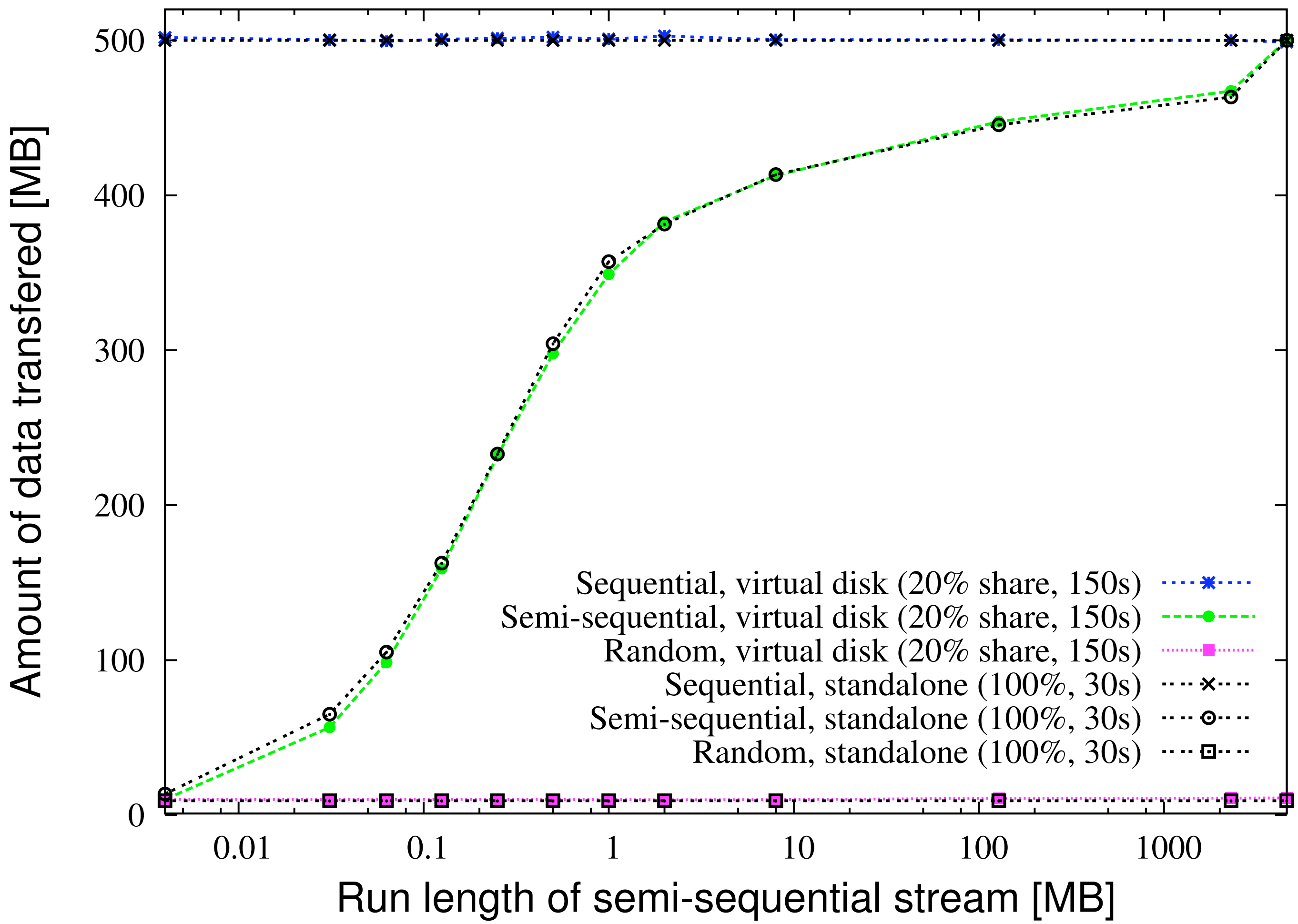


UC SANTA CRUZ

ISSDM

Guaranteeing throughput

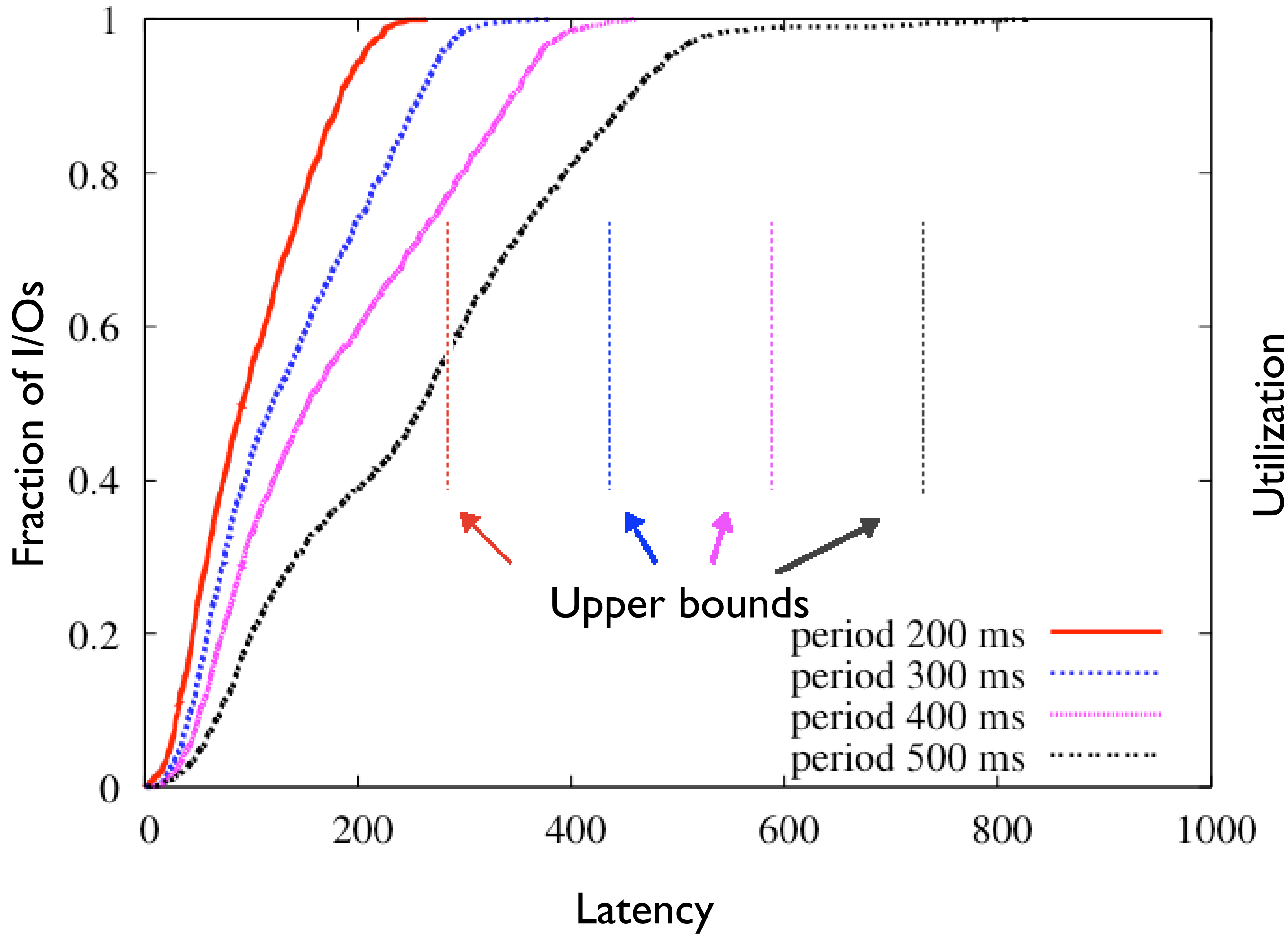
- Throughput is fully determined by reservation & workload
- Virtual disk are completely isolated from each other



Each virtual disk reserves 20% with 1 second granularity

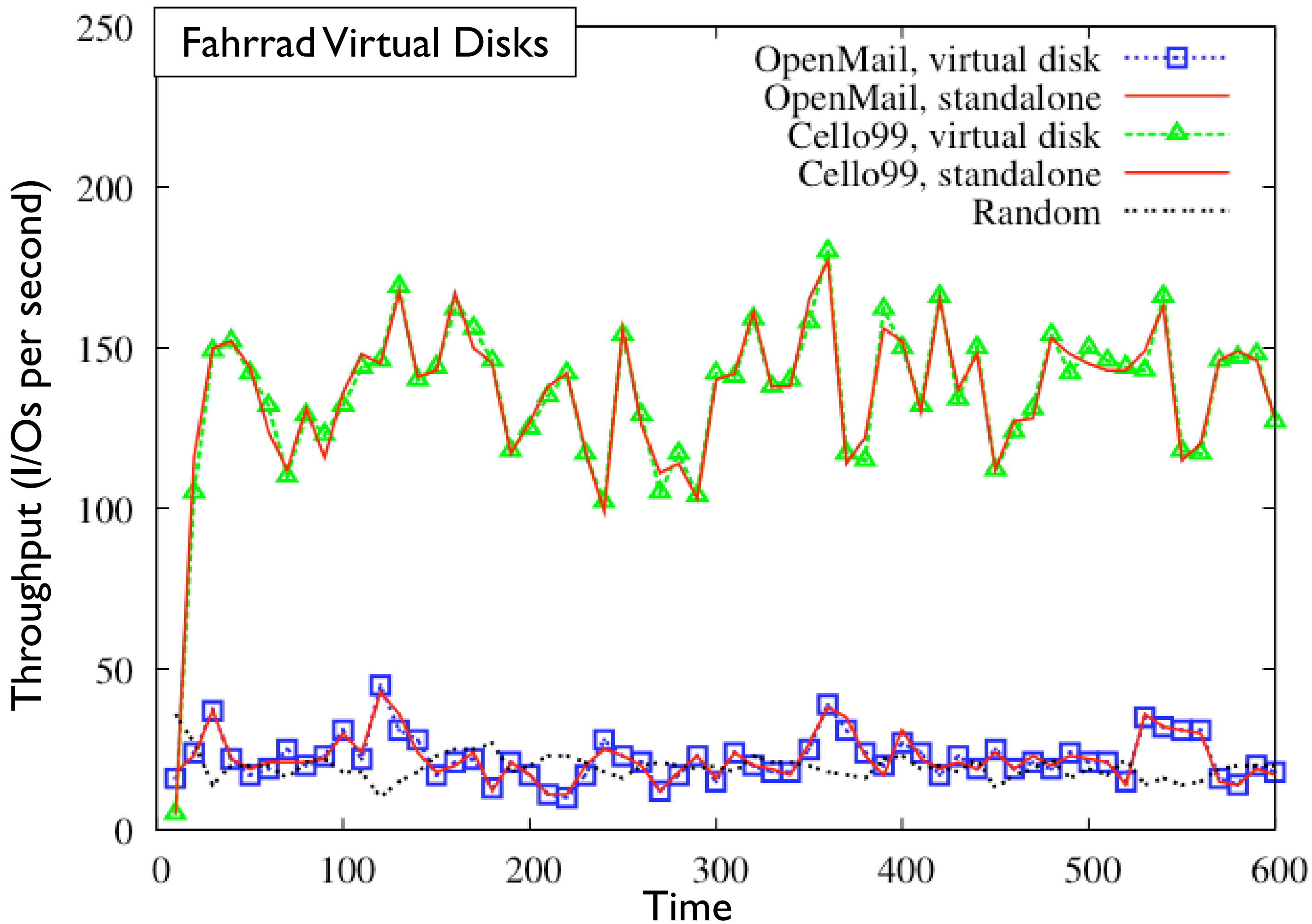
Guaranteeing latency

- Latency is bounded by deadlines

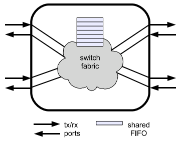


Fahrrad virtual disks work

- Fahrrad Virtual Disks provide Cello99 and OpenMail performance very close to standalone
- Cello99 and OpenMail virtual disks share the system with random background stream.

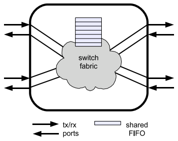


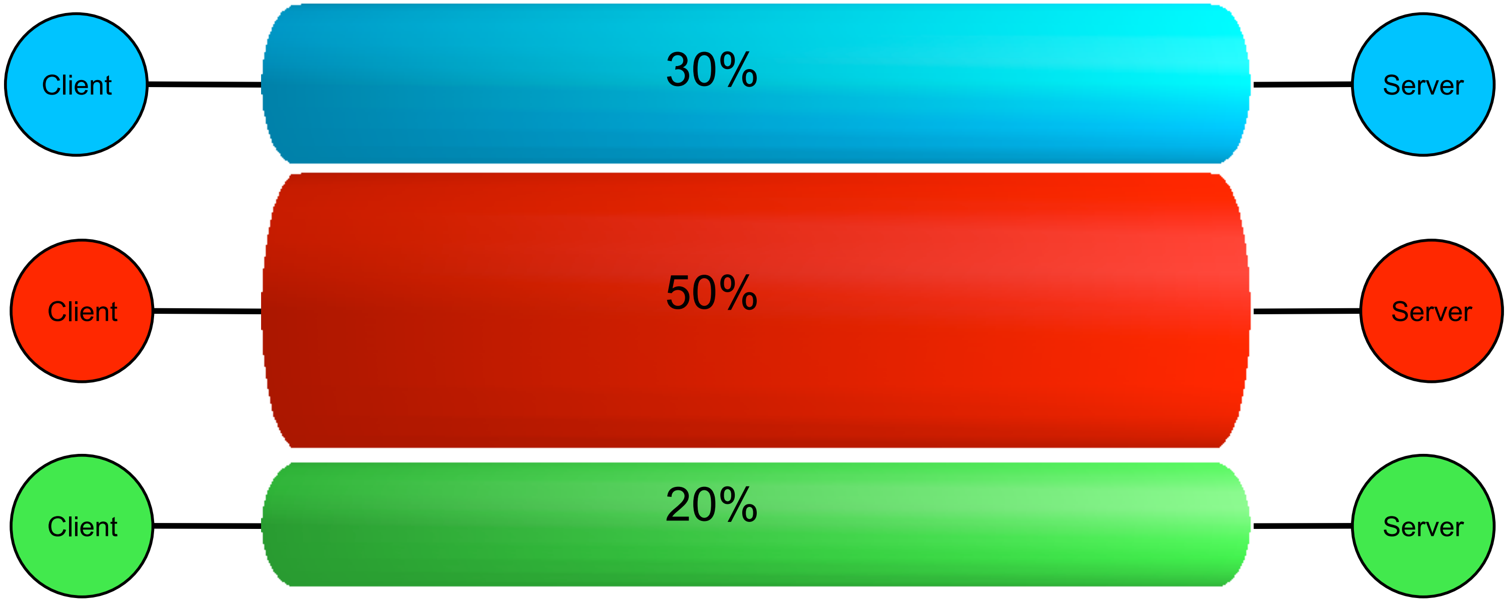
Guaranteeing storage network performance



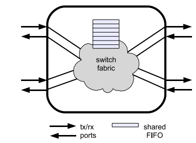
- Goals
  - Hard and soft performance guarantees
  - Isolation between I/O streams
  - Good I/O performance
- Challenging because network I/O is:
  - Distributed
  - Non-deterministic (due to collisions or switch queue overflows)
  - Non-preemptable
- Assumption: closed network

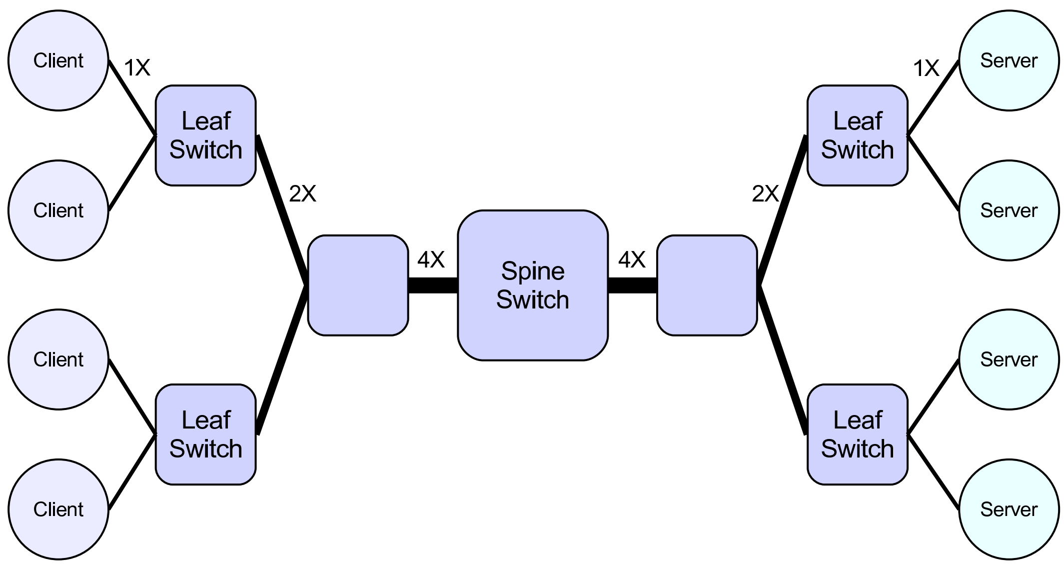
What we want

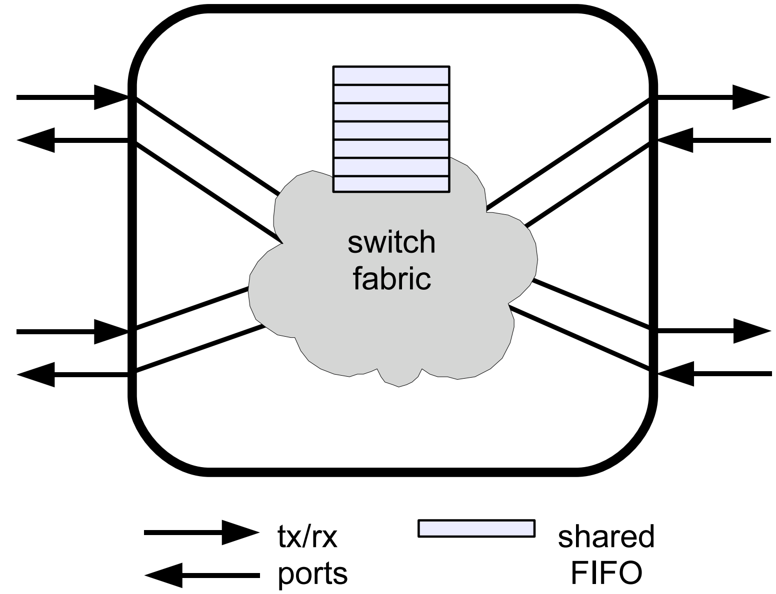

switch
fabric
tx/rx
ports
shared
FIFO


Client
30%
Server
Client
50%
Server
Client
20%
Server

What we have



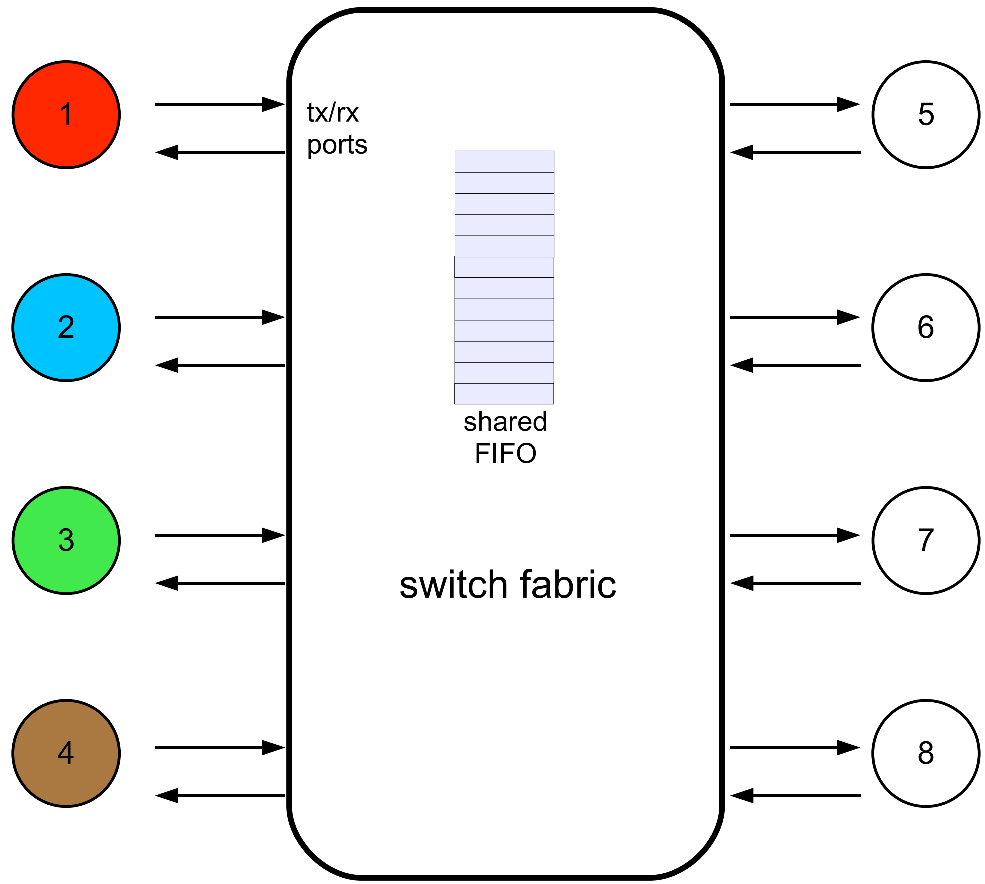




- Switched fat tree w/full bisection bandwidth
- Issue 1: Capacity of shared links
- Issue 2: Switch queue contention

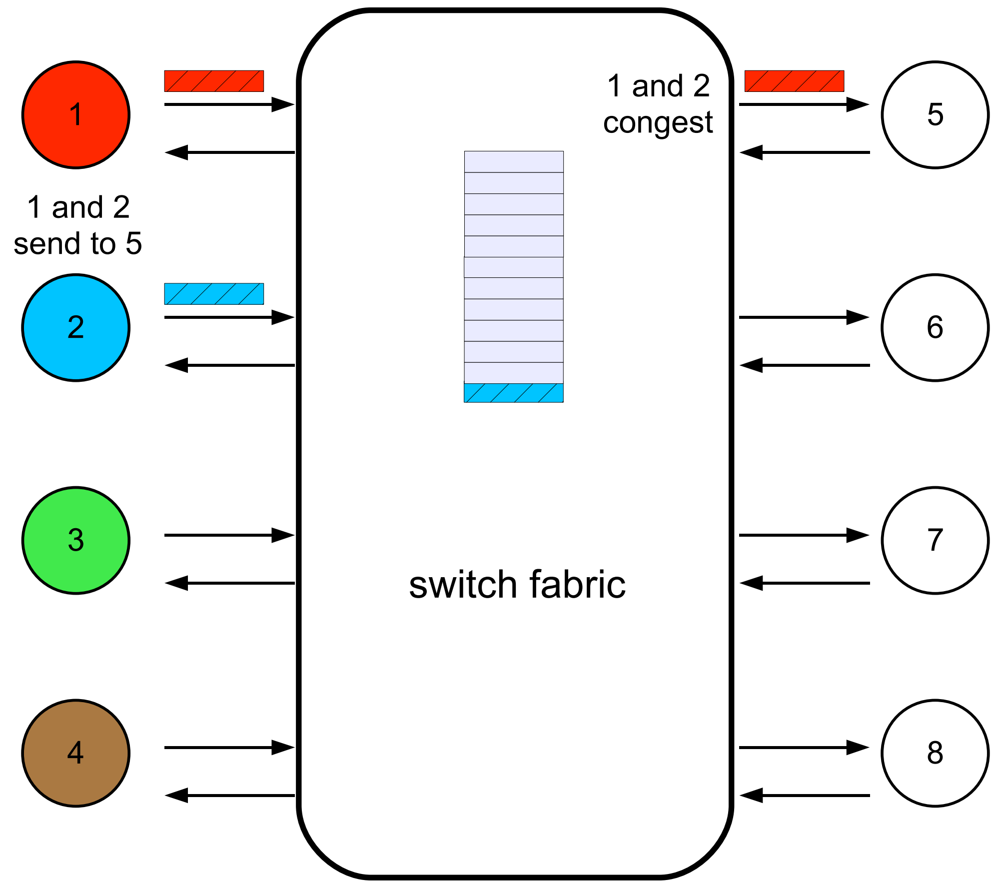
Congestion in a simple switch model

- Each transmit port on the switch is a collision domain



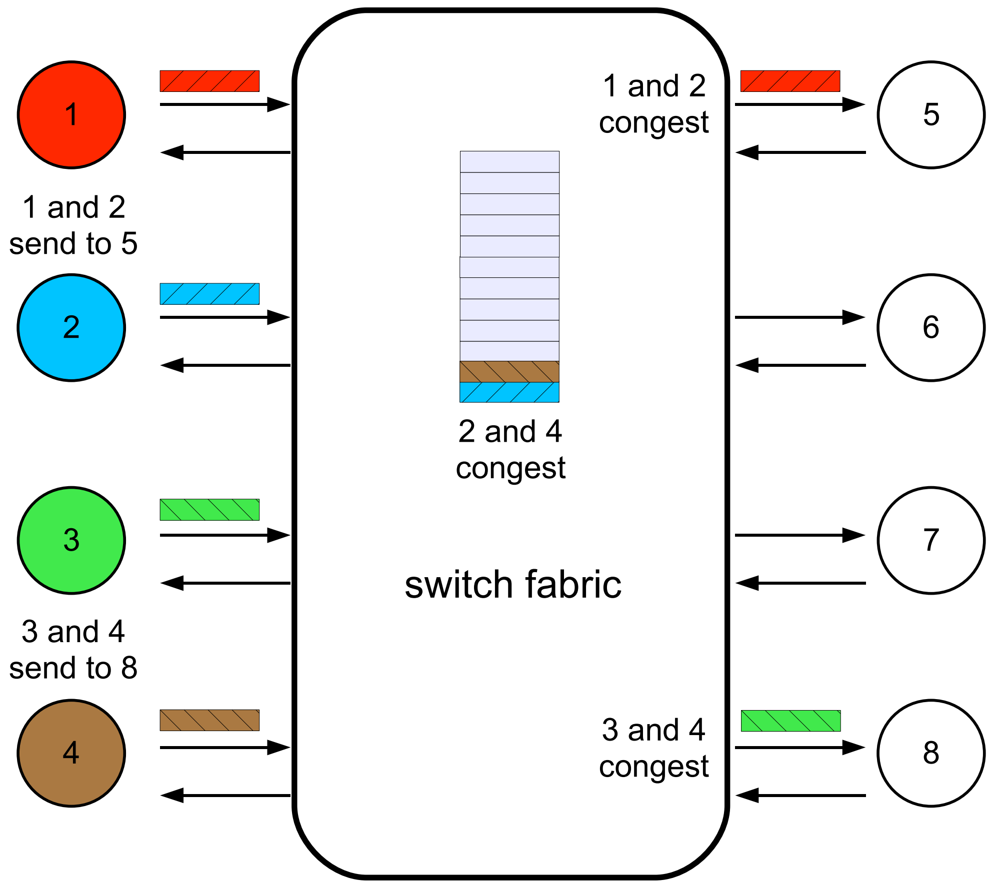
Congestion in a simple switch model

- One of the packets arriving at the same switch transmit port is delayed on the queue



Congestion in a simple switch model

- Delayed packets from unrelated streams affect each other on the queue

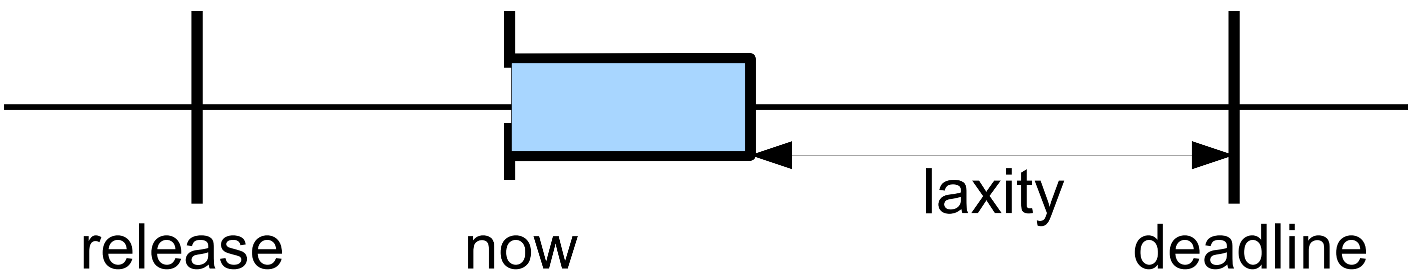


RADoN

- Each reservation has a *network share* (*utilization*) and a *time granularity (period)*

- Flow control: throttle senders
  - Execution time (per period) e = utilization / period
  - Budget in packets m = e / packets_per_second

- Congestion control: avoid switch contention by adjusting wait time between packets
  - Percent budget %budget = (1 - %laxity) = e/(d-t)
  - Packet wait time $w = w_{min}$ / %budget
  - Size change $w\Delta = -|w_i - w_{min}|/2$
  - New wait time $w_{i+1} = \min(w_{max}, \max(w_{min}, w\Delta))$

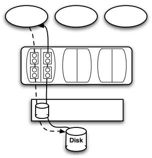


Tim Kaldewey, Andrew Shewmaker, Carlos Maltzahn, Theodore Wong, and Scott Brandt, "*RADoN: QoS in Storage Networks*", WIP Session of FAST 2008.

UC SANTA CRUZ

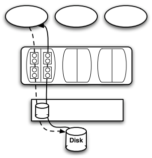
ISSDM

Buffer management for I/O guarantees



- Goals
  - Hard and soft performance guarantees
  - Isolation between I/O streams
  - Improved I/O performance
- Challenging because:
  - Buffer is space-shared rather than time-shared
    - Space limits time guarantees
  - Best- and worst-case are opposite of disk
  - Buffering affects performance in non-obvious ways

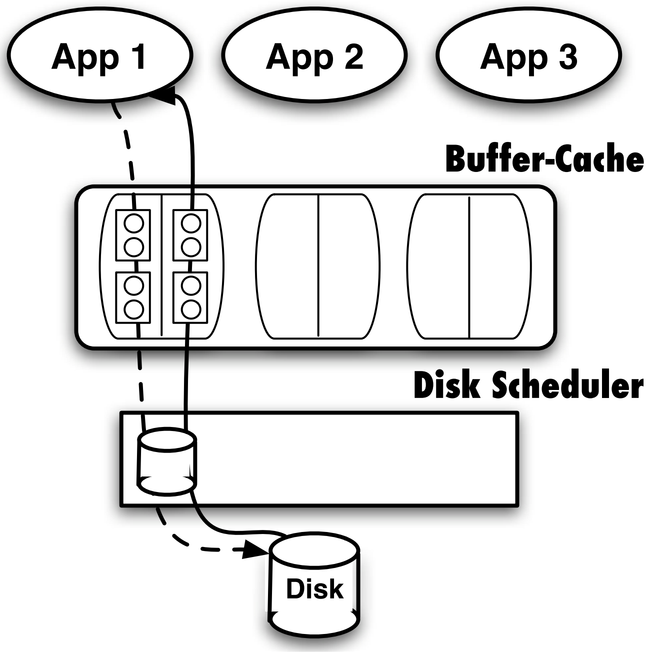
Buffering roles in storage servers



- Decoupling
  - Allows sender and receiver to operate asynchronously
- Speed matching
  - Allows slower and faster devices to communicate
- Traffic shaping
  - Shapes traffic to optimize performance of interfacing devices

Radium

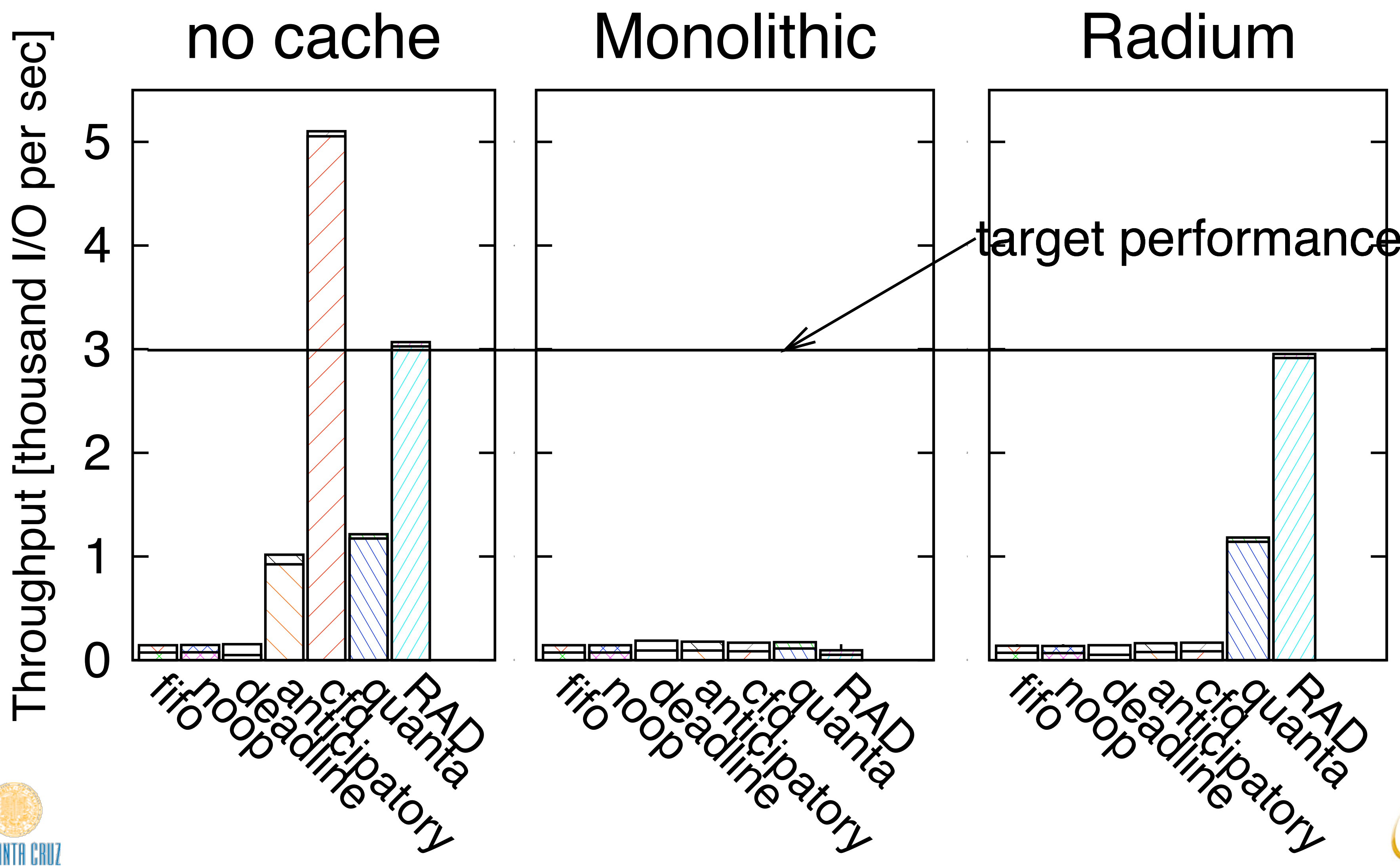
- I/O into and out of buffer have *rates* and *time granularities (periods)*
- Partition buffer space based on I/O characteristics and performance requirements
- Cache policies enhance performance within constraints determined by I/O requirements
  - Use slack to prefetch reads and delay writes



Roberto Pineiro, Kleoni Ioannidou, Carlos Maltzahn, and Scott Brandt, "RAD-FLOWS: Buffering for Predictable Communication," RTAS 2011

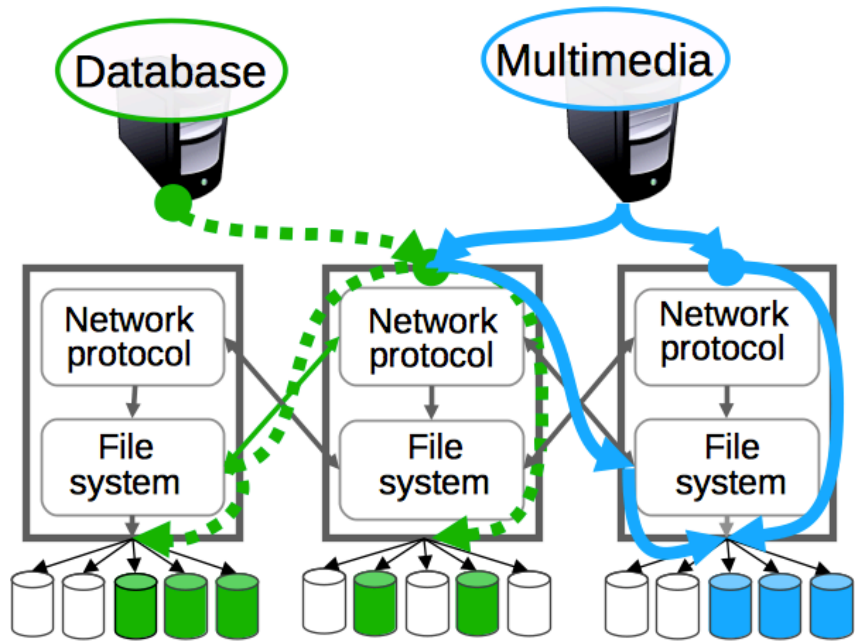
Managing combined workloads

Combined throughput of rand.(top) and seq.(bottom) workloads



Horizon

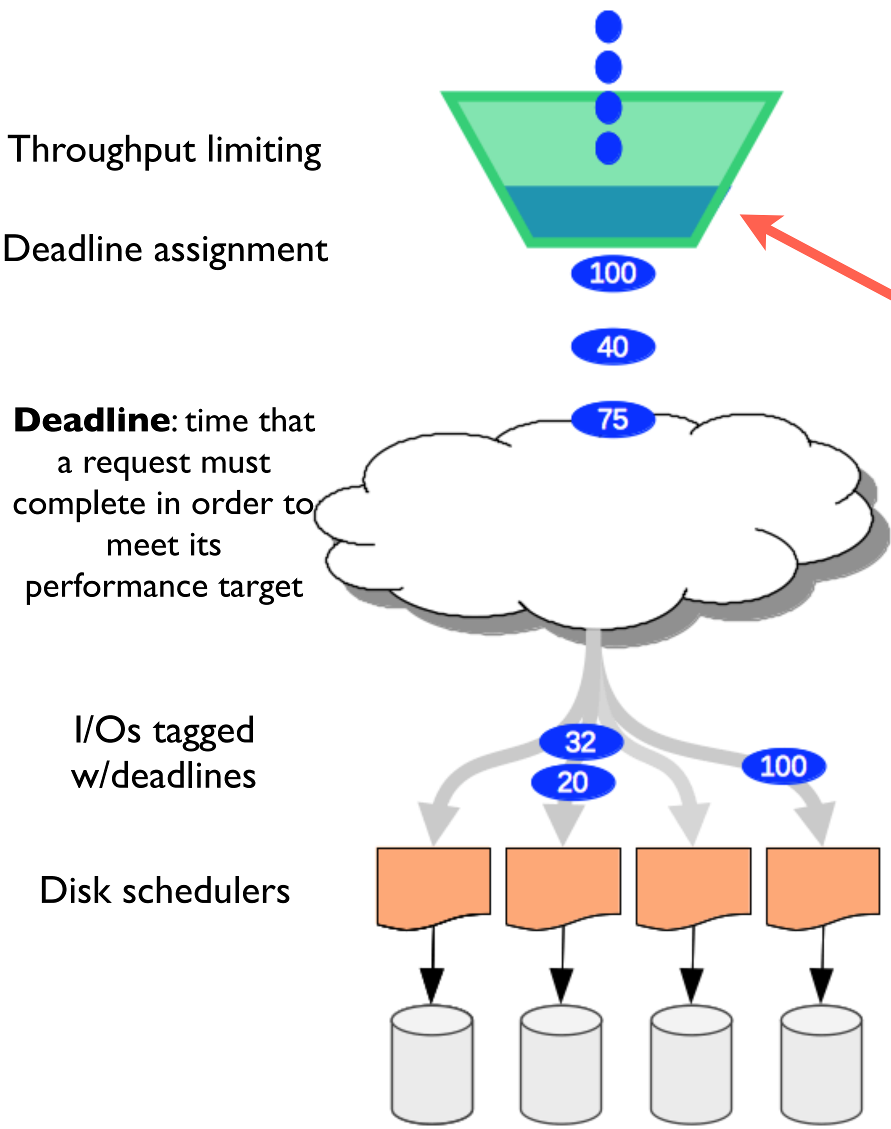
- Big storage systems are shared, have many disks, and application workloads compete and interfere
- Real distributed systems have
  - Different data layouts
  - Multiple data entry points
  - Different data paths
- Horizon goals
  - Meet performance targets
  - Fully utilize system resources
  - Not rely on reservations
  - Decentralized solution



Anna Povzner, Darren Sawyer, Scott A. Brandt, "Horizon: Efficient Deadline-Driven Disk I/O Management for Distributed Storage Systems," HPDC 2010. **Best Paper**

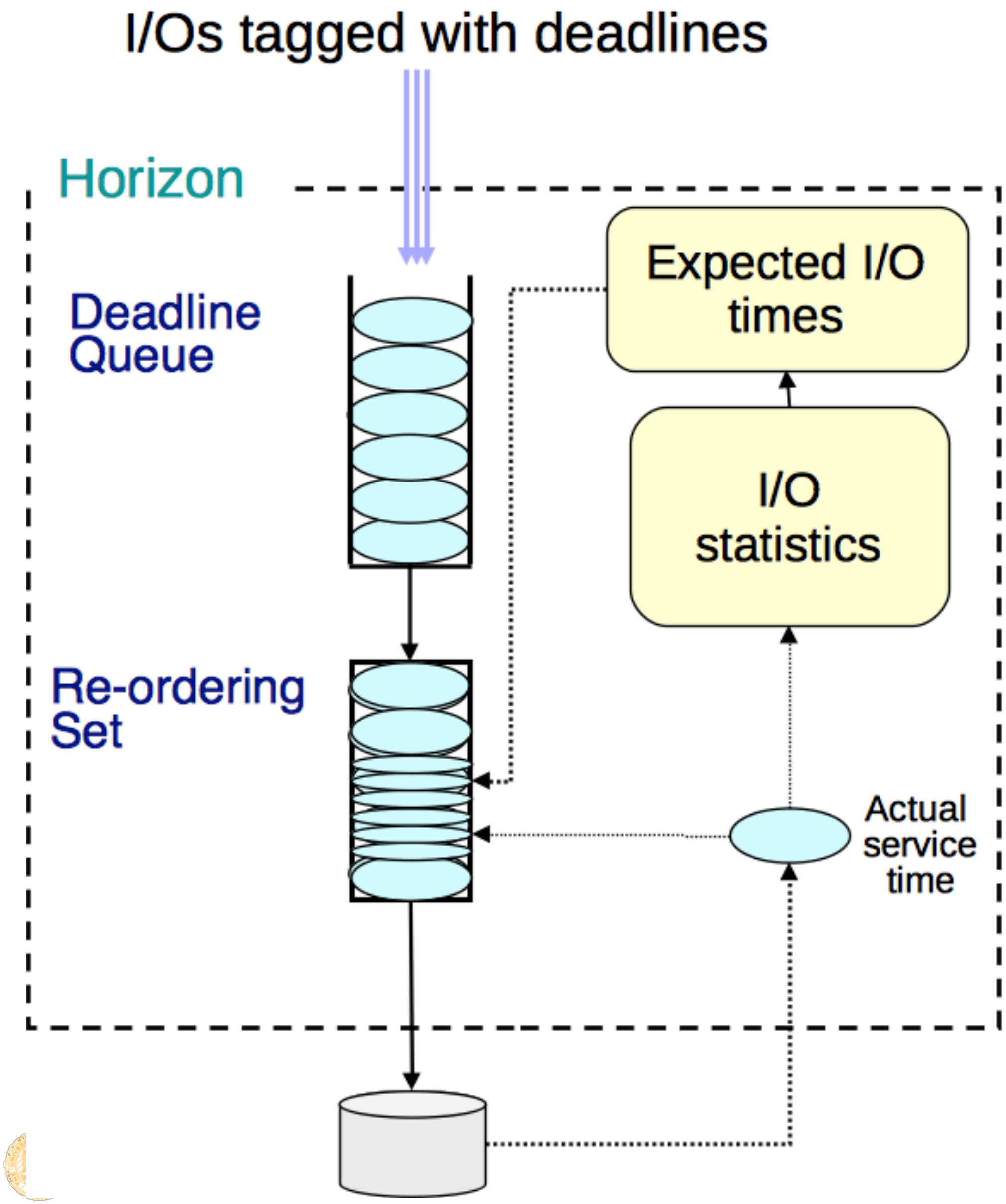
UC SANTA CRUZ

Multi-layered approach



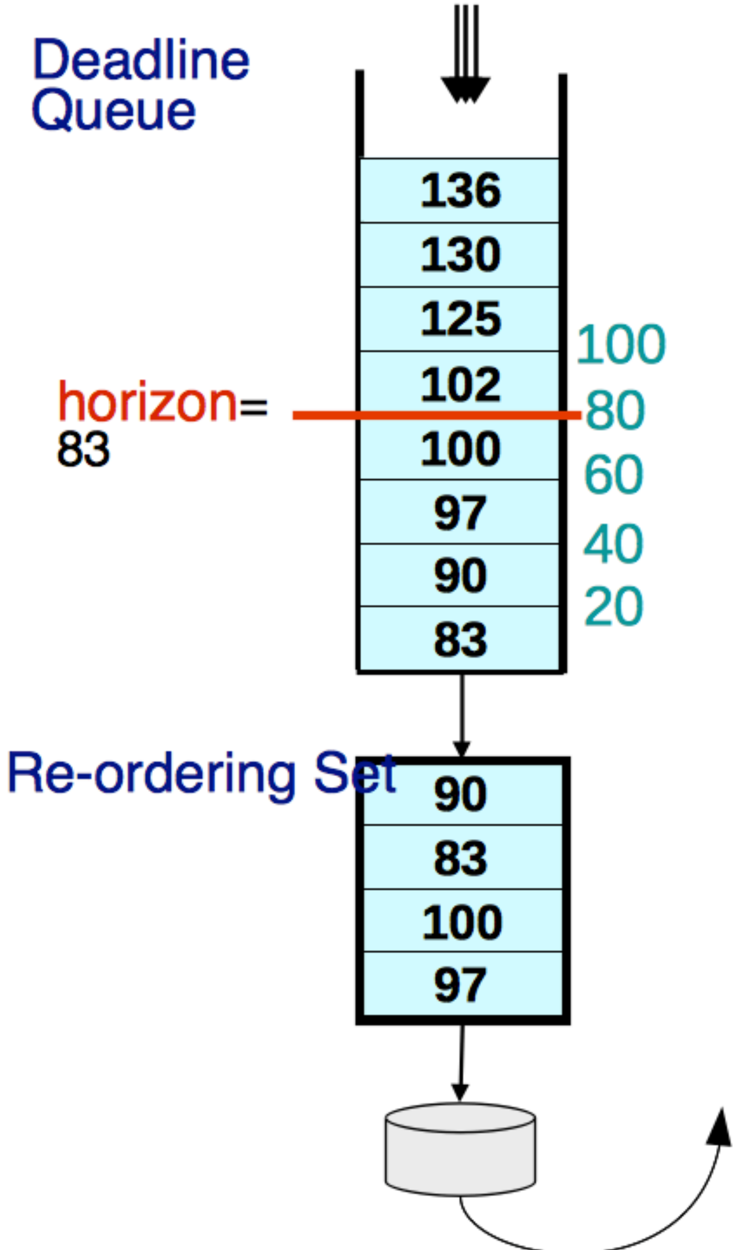
- Workloads specify performance targets
  - Throughput and latency
- Upper layer control mechanism
  - Throughput limiting
  - Deadline assignment based on throughput and latency targets
- Low-level disk schedulers
  - Meet individual request deadlines

Horizon disk scheduling



- Manage I/O in terms of disk time
- Estimate service times based on service time measurements
- Reorder requests within “slack time” before earliest deadline
- Adjust based on optimizations, overload, latency

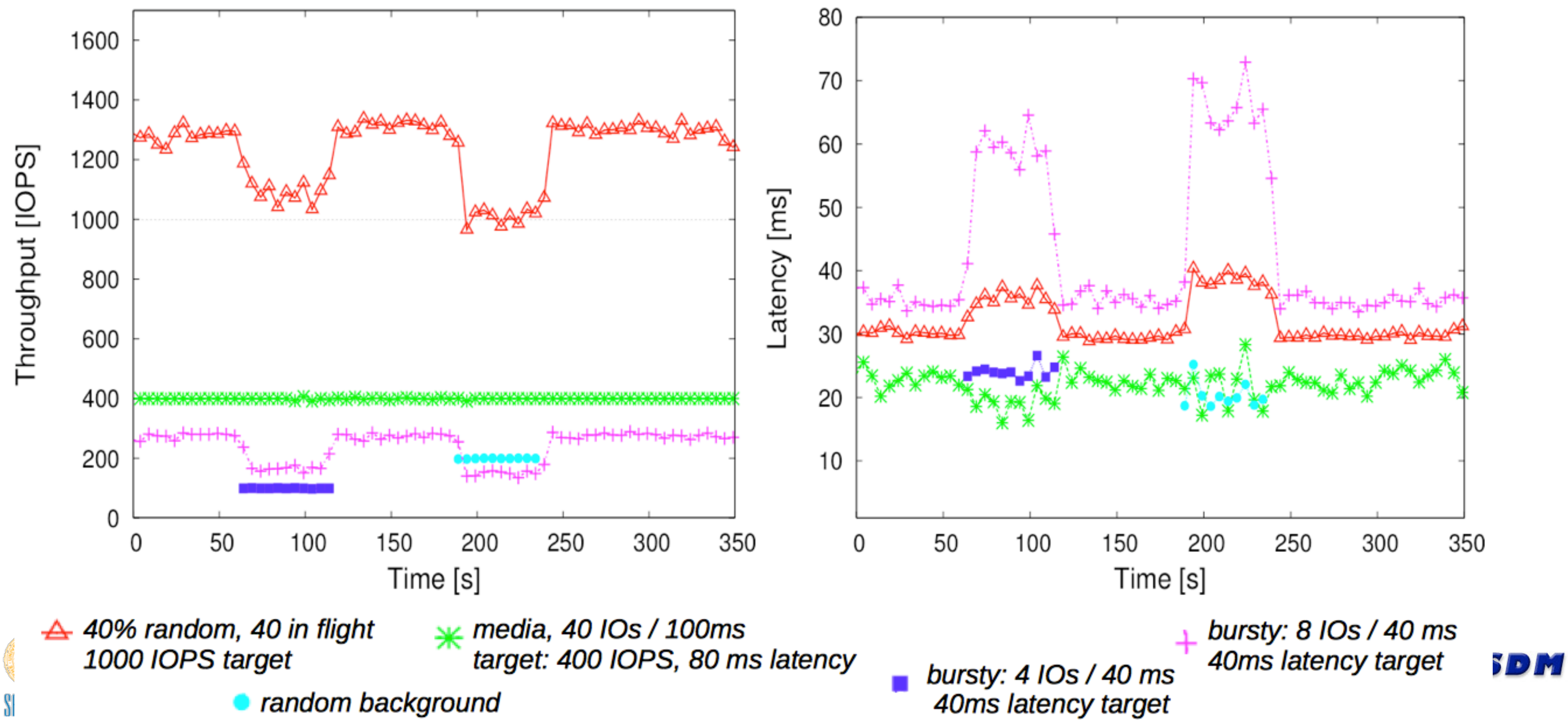
Horizon disk scheduling



- Horizon set to earliest deadline
- Reordering set = everything that will fit before horizon
- Execution times measured as requests complete
- Optimizations
  - Squeeze in more sequential I/Os
  - Use optimistic estimates
  - Increase reordering set (esp. under overload)
  - Increase device queue
    - Larger = better performance
    - Smaller = tighter deadlines

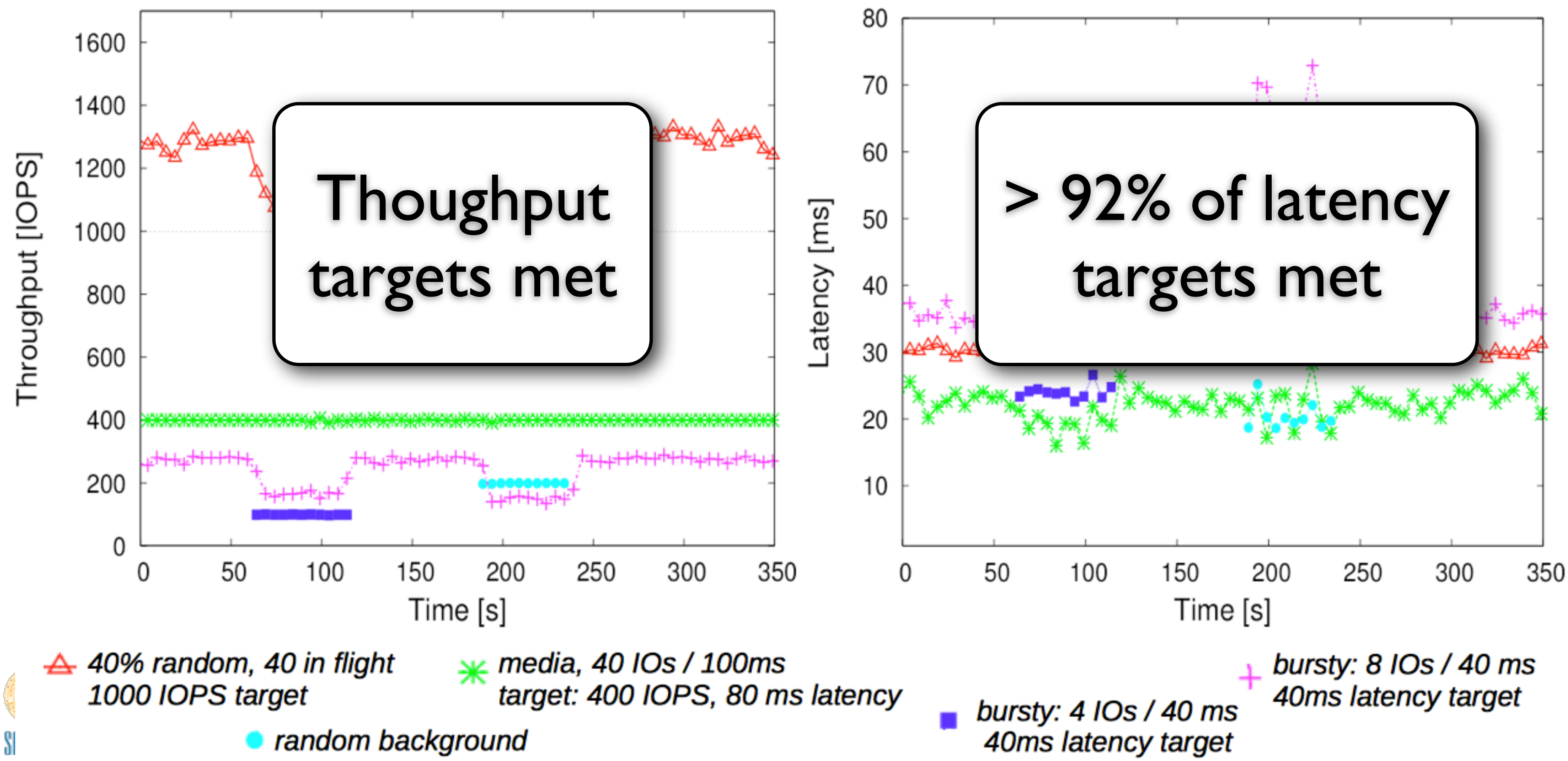
Horizon in use

- Implemented in NetApp's Data ONTAP (data from FAS3040)
- Performance targets associated with volumes
    - Control mechanism at FS entry point
    - Schedulers between RAID and disks



Horizon in use

- Implemented in NetApp's Data ONTAP (data from FAS3040)
- Performance targets associated with volumes
  - Control mechanism at FS entry point
  - Schedulers between RAID and disks



Conclusion

- I/O performance management is needed, challenging, and feasible
  - Many separate elements are involved
  - A unified approach is ideal
- RAD is the basis for a unified solution
  - CPU, disk, network, buffer cache, system
- It is in use in a commercial storage system
- More in the works

UC SANTA CRUZ

ISSDM